## Problem1

遍历所有的文件，以单词作为关键字，如果发现文件含有该关键字就将文件名加在该关键字所指向的数组里。

```
for line in sys.stdin:
    args = json.loads(line)
    fileName = args[0]
    value = args[1]
    words = value.split()
    for w in words:
        mr.emit_intermediate(w, fileName)
```

```
for line in sys.stdin:
    json_info = json.loads(line)
    for key in json_info:
        fileList = []
        fileNames = json_info[key]
        for fileName in fileNames:
            if fileName not in fileList:
                fileList.append(fileName)
        mr.emit((key, fileList))
```

## Problem2

在 mapper 里将每行输入的 id 作为字典的 key，将整个输入作为字典的值，然后将整个字典输出给 reducer。在 reducer 里如果发现两项的 id 相同并且一个是 order 一个是 item，就从字典里取出这两个值将其连接起来。

```
for line in sys.stdin:
    args = json.loads(line)
    order_id = args[1]
    value = args
    mr.emit_intermediate(order_id, value)
```

```
for line in sys.stdin:
        intermediate = json.loads(line)
        for key in intermediate:
            values = intermediate[key]
            global order
            for value in values:
                if value[0] == 'order':
                    order = value

            for value in values:
                if value[0] == 'line_item':
                    mr.emit((order + value))
```

## Problem3

在 mapper 里将人按照名字作为字典的关键字进行分类, 在 reducer 里遍历数据, 如果发现有两个人之间是朋友但是在其中一个人作为关键字的字典里没有另一个人的名字就将那个人的名字加在字典里，最后输出每个人的朋友数组的大小。

```
for line in sys.stdin:
        record = json.loads(line)
        name = record[0]
        mr.emit_intermediate(name, 1)

for line in sys.stdin:
        intermediate = json.loads(line)
        for key in intermediate:
            name = key
            list_of_values = intermediate[key]
            total = 0
            for v in list_of_values:
                total += v
            mr.emit((name, total))
```

## Problem4

在 mapper 里将输入全部变成键值对，在 reducer 里对 mapper 的输出进行整理，遍历所有关系对，对于每一个人，如果发现他有某个朋友但是自己却不是那个人的朋友就将此关系对输出。

```
for line in sys.stdin:
        friendship = json.loads(line)
        mr.emit_intermediate(friendship[0], friendship[1])
        mr.emit_intermediate(friendship[1], friendship[0])


for line in sys.stdin:
        intermediate = json.loads(line)
        for key in intermediate:
            person = key
            list_of_friends = intermediate[key]
            friendCount = {}
            for friend in list_of_friends:
                friendCount.setdefault(friend, 0)
                friendCount[friend] = friendCount[friend] + 1


            asymfriends = filter(lambda x : friendCount[x] == 1, friendCount.keys())


            for friend in asymfriends:
                mr.emit((person, friend))
```

## Problem5

在 mapper 里对每个 DNA 序列进行裁剪，然后把基因序列作为字典的 key，原来的 id 作为字典的值，输出到 reducer 里，在 reducer 里将 key 输出。

```
for line in sys.stdin:
        dnaseq = json.loads(line)
```

```
        seqId = dnaseq[0]
        nucleotide = dnaseq[1]
        trimmedNucleotide = nucleotide[:-10]
        mr.emit_intermediate(trimmedNucleotide, seqId)

for line in sys.stdin:
        intermediate = json.loads(line)
        for key in intermediate:
            trimmedNucleotide = key
            mr.emit(trimmedNucleotide)
```

## Problem6

在 mapper 里按照 a，b 矩阵来分类，输出到 reducer 里，在 reducer 里将所有 A 矩阵的输出合并起来，将 B 矩阵的输出合并起来，然后遍历两个集合，如果发现 A 矩阵项的纵坐标和 B 矩阵的横坐标相同就将两项相乘加入最终的结果里。

```
for line in sys.stdin:
        record = json.loads(line)
        maxI = 10
        maxJ = 10

        if record[0] == 'a':
            i = record[1]
            for j in range(maxJ + 1):
                mr.emit_intermediate(str((i, j)), record)
        elif record[0] == 'b':
            j = record[2]
            for i in range(maxI + 1):
                mr.emit_intermediate(str((i, j)), record)
        else:
            pass

for line in sys.stdin:
```

```
intermediate = json.loads(line)
for key in intermediate:
    values = intermediate[key]
    values = list(values)
    a_rows = filter(lambda x : x[0] == 'a', values)
    b_rows = filter(lambda x : x[0] == 'b', values)

    result = 0
    for a in a_rows:
        for b in b_rows:
            if (a[2]==b[1]):
                result += a[3] * b[3]

    # emit non-zero results
    if (result != 0):
        key = eval(key)
        mr.emit((key[0], key[1], result))
```